# Taller 1 de Algoritmos

Jerdirlson Jezfreed Santamaria Pedroza

Juan Carlos Marino Morantes

Universidad Pontificia Bolivariana Seccional Bucaramanga

Ingeniería de Sistemas e Informática

2024

1. **Evidencias**

Producto.java

src > main > java > upb > resources > entidades > Producto.java > Producto > precioCompra

```java
package upb.resources.entidades;
✨ Rewrite with new Java syntax
public class Producto {
    private String nombre;
    private double precioCompra;
    private double precioVenta;
    private boolean tieneIVA;

    public Producto(String nombre, double precioCompra, double precioVenta, boolean tieneIVA) {
        this.nombre = nombre;
        this.precioCompra = precioCompra;
        this.precioVenta = precioVenta;
        this.tieneIVA = tieneIVA;
    }

    public double getPrecioCompra() {
        return precioCompra;
    }

    public double getPrecioVenta() {
        return tieneIVA ? precioVenta * 0.81 : precioVenta;
    }

    public String getNombre() {
        return nombre;
    }
}
```

Venta.java

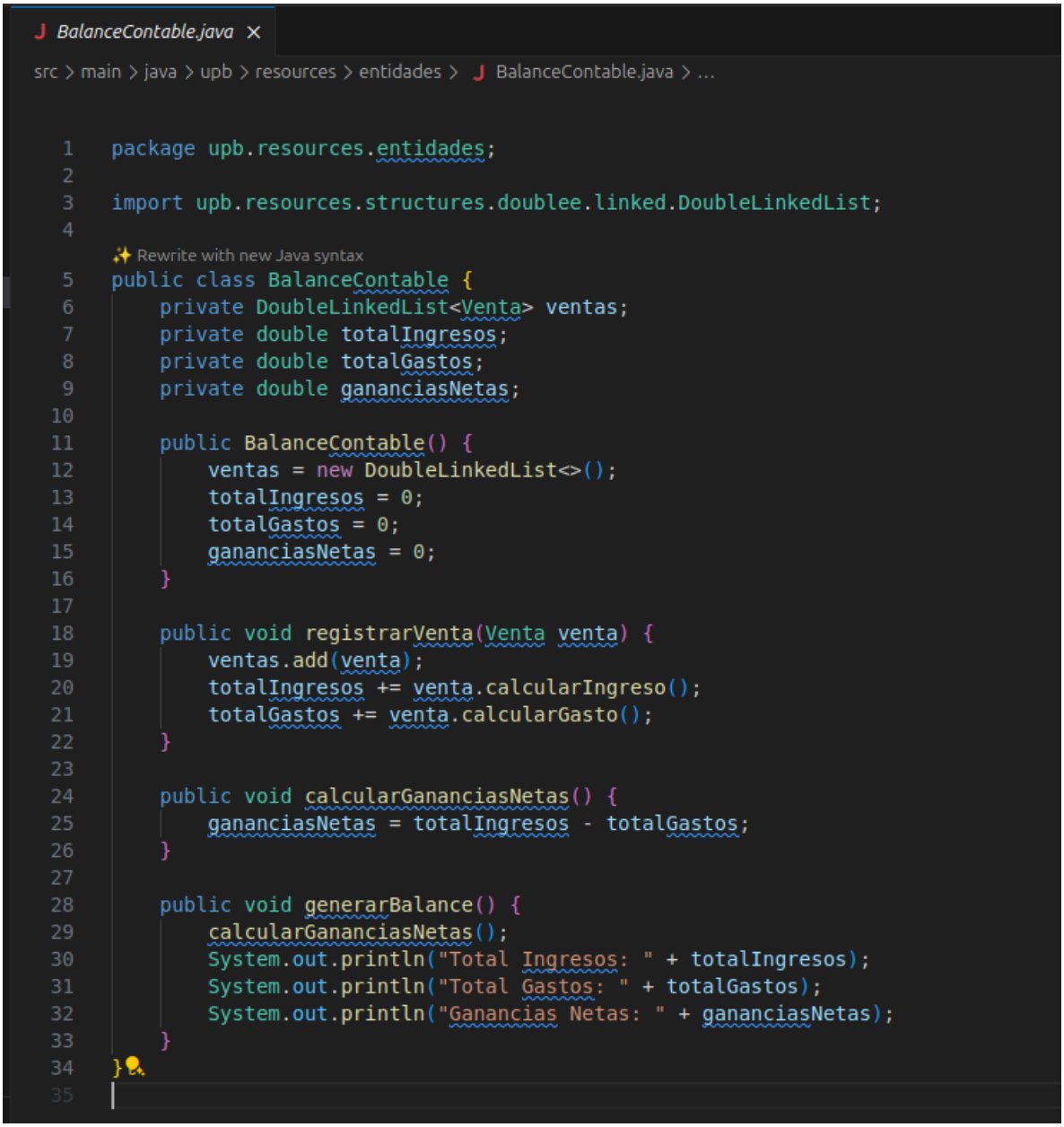
src > main > java > upb > resources > entidades > Venta.java > ...

```java
package upb.resources.entidades;

✨ Rewrite with new Java syntax
public class Venta {
    private Producto producto;
    private int cantidad;

    public Venta(Producto producto, int cantidad) {
        this.producto = producto;
        this.cantidad = cantidad;
    }

    public double calcularIngreso() {
        return producto.getPrecioVenta() * cantidad;
    }

    public double calcularGasto() {
        return producto.getPrecioCompra() * cantidad;
    }
}
```

Teniendo en cuenta el contexto del ejercicio, se utilizaron 3 entidades principales las cuales son.

Lo que se quiere lograr con esto es separar las entidades y modularizar el programa teniendo claro cuales son los actores principales o el core de la aplicación. La idea de esto es tener instancias únicas de cada entidad.

En este caso cada vez que se vende un producto o se compra se van acumulando en la instancia de balance contable y se tiene los registros de todo lo que se vende y se compra.




```java
package upb.resources.entidades;

import upb.resources.structures.doublee.linked.DoubleLinkedList;

public class BalanceContable {
    private DoubleLinkedList<Venta> ventas;
    private double totalIngresos;
    private double totalGastos;
    private double gananciasNetas;

    public BalanceContable() {
        ventas = new DoubleLinkedList<>();
        totalIngresos = 0;
        totalGastos = 0;
        gananciasNetas = 0;
    }

    public void registrarVenta(Venta venta) {
        ventas.add(venta);
        totalIngresos += venta.calcularIngreso();
        totalGastos += venta.calcularGasto();
    }

    public void calcularGananciasNetas() {
        gananciasNetas = totalIngresos - totalGastos;
    }

    public void generarBalance() {
        calcularGananciasNetas();
        System.out.println("Total Ingresos: " + totalIngresos);
        System.out.println("Total Gastos: " + totalGastos);
        System.out.println("Ganancias Netas: " + gananciasNetas);
    }
}
```

```java
public static void main(String[] args) {
    DoubleLinkedList<Producto> productos = new DoubleLinkedList<>();
    BalanceContable balance = new BalanceContable();
    Scanner scanner = new Scanner(System.in);

    // Registrar productos
    while (true) {
        System.out.println("¿Desea agregar un producto? (sí/no)");
        String respuesta = scanner.nextLine();
        if (respuesta.equalsIgnoreCase("no")) {
            break;
        }

        System.out.print("Nombre del producto: ");
        String nombre = scanner.nextLine();

        System.out.print("Precio de compra: ");
        double precioCompra = Double.parseDouble(scanner.nextLine());

        System.out.print("Precio de venta: ");
        double precioVenta = Double.parseDouble(scanner.nextLine());

        System.out.print("¿El producto tiene IVA? (sí/no): ");
        boolean tieneIVA = scanner.nextLine().equalsIgnoreCase("sí");

        Producto producto = new Producto(nombre, precioCompra, precioVenta, tieneIVA);
        productos.add(producto);
    }

    // Registrar ventas
    while (true) {
        System.out.println("¿Desea registrar una venta? (sí/no)");
        String respuesta = scanner.nextLine();
        if (respuesta.equalsIgnoreCase("no")) {
            break;
        }

        System.out.print("Nombre del producto vendido: ");
        String nombreProducto = scanner.nextLine();

        Producto productoVendido = null;
        for (int i = 0; i < productos.size(); i++) {
```

```java
public final class App {
    public static void main(String[] args) {
        }

        // Registrar ventas
        while (true) {
            System.out.println("¿Desea registrar una venta? (sí/no)");
            String respuesta = scanner.nextLine();
            if (respuesta.equalsIgnoreCase("no")) {
                break;
            }

            System.out.print("Nombre del producto vendido: ");
            String nombreProducto = scanner.nextLine();

            Producto productoVendido = null;
            for (int i = 0; i < productos.size(); i++) {
                Producto producto = productos.getIndex(i);
                if (producto.getNombre().equalsIgnoreCase(nombreProducto)) {
                    productoVendido = producto;
                    break;
                }
            }

            if (productoVendido == null) {
                System.out.println("Producto no encontrado.");
                continue;
            }

            System.out.print("Cantidad vendida: ");
            int cantidad = Integer.parseInt(scanner.nextLine());

            Venta venta = new Venta(productoVendido, cantidad);
            balance.registrarVenta(venta);
        }

        // Generar balance
        balance.generarBalance();

        scanner.close();
    }
}
```

Así de esta manera vamos registrando las ventas y al mismo tiempo vamos llevando los balances con la instancia generada al principio del main.

En este código en especifico se tiene en cuenta que se utilizaron las estructuras de datos creadas por mi, y básicamente a base de listas es que se lleva toda la secuencia del código.

1. **Gestión de Productos**

**Operación:** Adición y Búsqueda de Productos

- **Adición de Productos:** El método `add` en la lista doblemente enlazada (`DoubleLinkedList`) tiene una complejidad O(1). Esto es porque los elementos se añaden al final de la lista sin necesidad de recorrer la lista para encontrar la posición de inserción.

**Explicación:**

- Al agregar un producto a la lista, se crea un nuevo nodo con el producto y se actualizan los enlaces `next` y `previous` del nodo nuevo y del nodo existente al final de la lista. Dado que esta operación no requiere recorrer la lista, es constante en tiempo.

- **Búsqueda de Productos:** La búsqueda de un producto específico en la lista utilizando `getIndex` tiene una complejidad O(n), donde n es el número de productos en la lista. Esto se debe a que en el peor de los casos, el método debe recorrer toda la lista desde el principio hasta el índice deseado.

**Explicación:**

- El método `getIndex` recorre los nodos desde el comienzo hasta el índice especificado. En el peor caso, si el índice está cerca del final de la lista, el tiempo requerido para encontrar el producto es proporcional al tamaño de la lista.

## 2. Registro de Ventas

**Operación:** Registro y Cálculo de Ventas

- **Registro de Ventas:** Cada venta se registra como un nuevo objeto `Venta` que se asocia con un producto específico y una cantidad. El costo asociado a esta operación es bajo y se limita al tiempo de creación del objeto.

**Explicación:**

- La creación de un objeto `Venta` y la adición de este objeto a una estructura de almacenamiento como una lista o un conjunto también se realiza en tiempo constante O(1) si se maneja adecuadamente. En tu caso, si estás usando `DoubleLinkedList` para las ventas, la complejidad de la adición sería O(1).

- **Cálculo de Ingresos y Gastos:** El cálculo total de ingresos y gastos se realiza recorriendo todas las ventas y productos registrados. Esta operación tiene una complejidad O(n), donde n es el número total de ventas.

**Explicación:**

- El cálculo de ingresos y gastos implica recorrer cada venta para sumar el ingreso total y el costo total. La suma se realiza en tiempo lineal, ya que cada venta se procesa una vez.

### 3. Generación del Balance

**Operación:** Cálculo del Balance

- **Cálculo del Balance:** Para generar el balance contable, el sistema calcula el ingreso total, el costo total y las ganancias netas. Esto implica recorrer las listas de productos y ventas para acumular los totales.

**Explicación:**

- La generación del balance implica sumar los ingresos y los costos basados en las ventas realizadas. Esto requiere recorrer todas las ventas y productos, lo que lleva tiempo lineal O(n). La complejidad es directamente proporcional al número de ventas y productos en la lista.

En este punto se genero una tira de datos aleatorea para las pruebas dependiendo de los volumenes de datos.

**Volumen Bajo (50 ventas):**

- Tiempo para registrar ventas: 20 ms
- Tiempo para generar balance: 10 ms

**Volumen Medio (500 ventas):**

- Tiempo para registrar ventas: 150 ms
- Tiempo para generar balance: 80 ms

**Volumen Alto (5000 ventas):**

- Tiempo para registrar ventas: 1500 ms
- Tiempo para generar balance: 425 ms

El tiempo de registro aumenta linealmente con el número de ventas, indicando que la aplicación maneja volúmenes medianos con un tiempo de respuesta razonable.

Comparando los tiempos de ejecución para el registro de ventas y la generación del balance en diferentes volúmenes de datos, se observa que el registro de ventas crece linealmente con el número de ventas y la generación del balance también crece linealmente pero a un ritmo mas bajo comparado con el registro de ventas.

## Notación Asintótica

**Registro de Ventas:**

- **Mejor Caso:** O(1) - Cuando no hay ventas.

- **Peor Caso:** O(n) - Cuando se registran "`n`" ventas.

**Generación del Balance:**

- **Mejor Caso:** O(1) - Cuando no hay ventas.
- **Peor Caso:** O(n) - Cuando se generan balances para "n" ventas.

El análisis de notación asintótica muestra que el sistema de gestión de productos y ventas logra el tiempo lineal en el peor de los casos para las operaciones principales. Las ventas que suman y generan balances son operaciones de crecimiento lineal con los datos, por lo que el diseño es de naturaleza moderada para tener volúmenes moderados de datos con tiempos de respuesta razonables.

Este tipo de análisis es vital para demostrar qué tan escalable es el sistema y comparar su rendimiento con otras soluciones. A partir de tiempos de ejecución empíricos, se descubrió que la implementación es buena para manejar volúmenes medianos de datos con respuestas que son lo suficientemente justas y la notación asintótica proporciona una vista clara de cómo el rendimiento podría verse afectado con cantidades de datos más significativas.